Assignment 2: Your Best Software Project

1. Describe your "learning machine" and how you'll create a virtuous cycle of learning that will continually improve your team's understanding of the user.

Items to consider	Your plan for Learning
a. What vehicles (observations, metrics, tests, etc.) will you and your team use to make learning a regular part of your agile iterations?	Design sprints to understand the core needs to our users and to identity what is valuable to them. Make use of prototypes to validate our assumptions about user behaviour. Hold sprint retrospectives to identify what we learnt during each sprint.
b. What metrics will you use to measure outcomes?	Determine concrete acceptance test criteria that are relevant for the product. Metrics should test value and usability.
c. Who will take the lead on designing that learning? Who will take the lead on sharing it with the team in a way that drives strong, interdisciplinary discussions and links to the job of Deciding?	It should be the responsibility of everyone on the team but the product owner should play the lead role. The scrum master can play an important role in sharing and encouraging learning among stakeholders.
d. How often and in what meetings/workshops will you deliver the above?	Design sprints in the beginning of the project and sprint retrospectives after each sprint.
e. How will you present your user stories and related items (storyboards, story maps, etc.) to drive the best possible discussions in your sprint planning meetings?	Decide on a theme for each sprint and present the user stories and related items that are most relevant and valuable to that theme. This helps to keep the focus on a particular narrative.
f. What about after the sprint starts? How will you work to tilt the working environment towards thinking about what makes sense vs. just creating output--those 'blue button' moments?	Share progress in daily standup meetings to keep the team up to date on what everyone is doing and focussed on the bigger picture. Encourage individuals to highlight blockers or concerns about what makes sense.
g. What are the challenges and focal points you see for linking to the job of Deciding and creating a culture of experimentation?	Overcoming the fear of failure and irrationally staying committed to the wrong path due to sunk cost fallacy.

2. Describe how you'll decide what to build, when.

Items to consider	Your plan for Deciding
a. How long will your iterations be--and why? What might be the impact of longer iterations? Shorter?	Iterations will be two weeks. This has been found to be a reasonable amount of time to focus on a single theme. Longer iterations are harder to estimate and increase the risk that you complete too much work before being able to

	pivot in a different direction. Shorter sprints mean you probably can't build a set of cohesive features in one iteration.
b. How will you groom the product backlog to increase the quality of your inputs and prioritize tasks? What inputs and what people will be most important to your pre-sprint backlog grooming sessions? Why?	Use story maps to Identify the high priority user stories. These should form the core of the backlog. Any grooming should take place among the lower priority stories as these do not add significant value to the overall narrative. Key inputs should be the learnings from design sprints as well as the direction of the Product owner, who is responsible for the vision of the product, guiding and prioritizing features based on what is known to be valuable to users.
c. What is most critical to manage flow across design, development, and testing? For example, how will you make sure testing isn't backloaded to the very end of the sprint?	WIP (Work in progress) limits can be implemented to prevent a backlog developing at any particular stage of development. When a stage reaches its limit, the team should stop work and focus on clearing the backlogged areas.
d. Which practices from XP, scrum, and kanban will you use to make the job of deciding more effective?	SCRUM: Product backlog, Planning meetings and timeboxed iterations to maintain focus. Use Kanban board to visualize progress and implement WIP limits. XP - Maintain an Informative workspace to give the whole team a view of the big picture.
e. How will you evaluate the quality of your decisions and think about how to improve them?	Regular retrospective meetings to review the outcome of tests and data collected from our metrics.

3. Describe how you'll build (and test) software.

Items to consider	Your plan for Building
a. How will you support the development and testing team? If it's an existing team, which practices (from XP, scrum, and kanban) do you use? Why those? How well do you think those are working and do you think there are any changes worth considering? If it's a new team, which practices (from XP, scrum, and kanban) will you discuss with the team? Why those?	Follow the SCRUM technique of releasing working, tested software after each sprint. Allowing the team to Review feedback from the team during SCRUM retrospectives and use the feedback to make improvements to the development process. For complex and high risk features, encourage the use of pair programming to improve quality, even if it means a loss of productivity in the short team. Encourage the automation of the testing process to enable continuous integration.

b. How will you frame the 'win' on the practices you suggest for developers? Testers? Designers? (Choose whichever roles you currently have on the team.)	Developers: More value delivered with less effort (less time wasted developing features which will never be used). Automated testing and continuous integration means less risk when refactoring, therefore more agility. Testers: Clear focus on user stories means acceptance criteria are easier to identify and shared narrative means less disagreement among different roles. Short test cycles means problems are addressed early when its fresh in everyones minds, rather than at the end of the project.
c. How will you discuss the best way to run testing and decide who on the team will do what?	Aim should be to reduce handover overhead between development and testers. Encourage collaboration between dev and tests roles throughout the sprint. Ensure that conversations are driven by user stories to keep everyone on the same page.
d. How would you like to evolve the process of testing and deployment over time? What do you see as the first few steps?	Move away from siloed approach where development and testing is seperate. Automate as much as possible, starting with testing and then move to continuous deployment model where
e. How will you create slack to allow the team to avoid or reduce technical debt?	Evaluate technical debt on a regular basis, try to allocate 10% of each sprint to addressing this debt.

4. Describe how you will manage software development.

Items to consider	Your plan for Managing
a. What are the top three things you can do in your role to foster and contribute to a self-organizing team?	- Encourage a culture of trust and shared responsibility. - Reduce handovers by create opportunities for self service. - Allow teams to choose processes that work for them
b. What is the role of the retrospective for you? What agenda will you use? How will you tie the results of the retrospective back to the job of learning?	We want to learn about what went well and what can still be improved. Also we want to get feedback to the whole team in an open and non judgemental environment. Agenda should be flexible and aimed at honest reflexion. Tie results to learning by itemizing points for tracking and review these in the new retrospective to evaluate if progress has been made.